Licence d' Informatique - IUP2 Miage 2000 – 2001
LOGIQUE Feuille de **Examen** (Correction)

EXAMEN de LOGIQUE, 22 janvier 2001

Les documents sont autorisés. Le barème est donné à titre indicatif, et il est susceptible d'être modifié. Tout résultat prouvé en cours ou TD peut être utilisé, sans besoin de le démontrer de nouveau. Soyez clairs et concis.

Exercice 1
Barème indicatif : 3 points.

Utiliser le calcul des séquents **G** pour prouver le séquent

$$\forall x \exists y(\neg p(x) \ \wedge \ p(y)) \vdash$$

Vous avez le droit d'utiliser les règles de **G** pour $\exists$ données en TD. Que pouvez vous conclure sur la sémantique de la formule $\forall x \exists y(\neg p(x) \ \wedge \ p(y))$?

Correction : *Dans la preuve ci-dessous, v, z, w sont des variables.*

$$
\begin{array}{cl}
\hline
\forall x\exists y(\neg p(x) \wedge p(y)),\ p(w),\ p(v) \vdash p(v),\ p(z) & Axiome \\
\hline
\forall x\exists y(\neg p(x) \wedge p(y)),\ \neg p(v),\ p(w),\ p(v) \vdash p(z) & \neg\ g \\
\hline
\forall x\exists y(\neg p(x) \wedge p(y)),\ \neg p(v) \wedge p(w),\ p(v) \vdash p(z) & \wedge\ g \\
\hline
\forall x\exists y(\neg p(x) \wedge p(y)),\ \exists y(\neg p(v) \wedge p(y)),\ p(v) \vdash p(z) & \exists\ g \\
\hline
\forall x\exists y(\neg p(x) \wedge p(y)),\ p(v) \vdash p(z) & \forall\ g \\
\hline
\forall x\exists y(\neg p(x) \wedge p(y)),\ \neg p(z),\ p(v) \vdash & \neg\ g \\
\hline
\forall x\exists y(\neg p(x) \wedge p(y)),\ \neg p(z) \wedge p(v) \vdash & \wedge\ g \\
\hline
\forall x\exists y(\neg p(x) \wedge p(y)),\ \exists y(\neg p(z) \wedge p(y)) \vdash & \exists\ g \\
\hline
\forall x\exists y(\neg p(x) \wedge p(y)) \vdash & \forall\ g
\end{array}
$$

Puisque **G** *est correct, le séquent $\forall x \exists y(\neg p(x) \ \wedge \ p(y)) \vdash$ est valide, donc sa formule associée, c.à. d $(\forall x \exists y(\neg p(x) \ \wedge \ p(y))) \to \mathcal{F}$ est valide et $\forall x \exists y(\neg p(x) \ \wedge \ p(y))$ est insatisfiable.*

Exercice 2
Barème indicatif : 6 points.

Soit un langage avec deux symboles de prédicat binaires $<$ et $=$ et une fonction binaire $*$. On considère deux structures de ce langage :

- La structure $N = \langle \mathbb{N}, F_N, R_N \rangle$ où $\mathbb{N}$ est l'ensemble des nombres naturels (0 inclus), F_N est l'ensemble qui contient seulement la fonction de multiplication sur les nombres naturels, et R_N contient les deux relations “est inférieur à” et “est égal à” sur les entiers. L'interprétation I_N associe à $*$ la multiplication, à $<$ la relation “est inférieur à” et à $=$ la relation “est égal à”.

- La structure $R = \langle \mathbb{R}, F_R, R_R \rangle$ où $\mathbb{R}$ est l'ensemble des nombres réels, F_R est l'ensemble qui contient seulement la fonction de multiplication sur les nombres réels, et R_R contient les deux relations “est inférieur à” et “est égal à” sur les réels. L'interprétation I_R associe à $*$ la multiplication, à $<$ la relation “est inférieur à” et à $=$ la relation “est égal à”.

Pour chacune des formules suivantes dire si l'interprétation I_N en est un modèle. Même question pour I_R. Donner à chaque fois une **brève** justification.

1. $\forall x \exists y (x < y \vee x = y)$
2. $\exists x \forall y (x < y \vee x = y)$
3. $\forall x \forall y \Big(x < y \rightarrow \exists z (x < z \wedge z < y) \Big)$
4. $\forall x \forall y \exists z \forall v (z < v \leftrightarrow x < v \wedge y < v)$
5. $\forall x \exists x_1 \exists x_2 (x = x_1 * x_2 \wedge x_1 < x \wedge x_2 < x)$

Correction :

1. *Vrai dans les deux structures.*
2. *Vrai dans $\mathbb{N}$, faux dans $\mathbb{R}$.*
3. *Faux dans $\mathbb{N}$, vrai dans $\mathbb{R}$.*
4. *Vrai dans les deux structures (z est le maximum de x et y).*
5. *Faux dans les deux: Prendre $x = 0$ alors un des deux facteurs de 0 est 0.*

Exercice 3

Barème indicatif : 8 points.

Définition 1

1. *Une clause (du calcul des prédicats du premier ordre) $L_1 \vee ... L_n$ $(n \geq 1)$ est dite négative si et seulement si chaque littéral L_i a $\neg$ comme opérateur principal. Par exemple, $\neg r(a, x) \vee \neg q(y)$ est une clause négative, par contre $\neg r(a, x) \vee q(y)$ ne l'est pas.*
2. *Une clause (du calcul des prédicats du premier ordre) $L_1 \vee ... L_n$ est dite définie si et seulement si un et un seul littéral L_i est une formule atomique et les autres (éventuels) littéraux L_j ont $\neg$ comme opérateur principal. Par exemple, $\neg r(a, x) \vee q(y)$ est une clause définie , par contre $\neg r(a, x) \vee \neg q(y)$ ne l'est pas.*
3. *Soit BUT une clause négative, soit E un ensemble de clauses. Une réfutation (dans le système de preuve par résolution) de l'ensemble de clauses $BUT \cup E$ est dite être une réfutation* **linéaire input** *de point de départ BUT et de input E si et seulement si :*
   (a) *Elle utilise seulement la règle d'inférence RES, jamais la règle $FACT$.*
   (b) *L'inférence la plus haute (i.e. la plus lointaine de la racine) a la forme*
   $$\frac{BUT \quad cl}{G} \; RES$$
   où G est une clause quelconque et cl une clause quelconque de E ;
   (c) *Toute autre inférence est de la forme*
   $$\frac{G \quad cl}{H} \; RES$$
   où cl est une clause quelconque de E et G et H des clauses quelconques.

(d) La racine de l'arbre est $\emptyset$.

NB : on peut remarquer que cette définition implique que la réfutation aura alors une forme de **peigne**.

Dans cet exercice, $hanoi, move, append$ sont des symboles de prédicats, $cons, to, succ$ sont des symboles de fonction, $a, b, c, 0, nil$ des constantes, $A, B, C, M, N, Output, Ou1, Ou2, L1, L2, L3, Z$ sont des variables.
Les questions suivantes sont indépendantes l'une de l'autre.

1. Soit $BUT_1 = \neg hanoi(succ(0), Z)$, soit E_1 l'ensemble de clauses :

   (a) $hanoi(N, Output) \ \vee\ \neg move(a, b, c, N, Output)$

   (b) $move(A, B, C, succ(M), Output) \vee \neg move(A, C, B, M, Ou1) \vee \neg move(C, B, A, M, Ou2) \ \vee\ \neg append(Ou1, cons(to(A, B), Ou2), Output)$

   (c) $move(A, B, C, 0, nil)$

   (d) $append(cons(A, L1), L2, cons(A, L3)) \ \vee\ \neg append(L1, L2, L3)$

   (e) $append(nil, L1, L1)$

   Donner une réfutation linéaire input, de départ BUT_1 et de d'input E_1, de $BUT_1 \cup E_1$

2. Dans cette question et dans la suivante la seule constante est a, f et g sont des symboles de fonction, r et s des symboles de prédicats.
   Soit $BUT_2 = \neg s(v)$, soit E_2 l'ensemble contenant les clauses (a) et (b) suivantes :

   (a) $r(x, a, f(z)) \ \vee\ r(g(w), w, f(w)) \ \vee s(a)$

   (b) $\neg r(y, v, f(a)) \vee\ \neg r(g(a), v, f(v))$

   NB : Pour cette question, on pourra s'intéresser aux nombre de fois où le symbole de prédicat r apparaît dans les formules que l'on considère.

   - Existe t-il une réfutation qui ne soit pas linéaire input de l'ensemble de clauses $BUT_2 \cup E_2$? Si oui, en donner une, sinon, justifiez votre réponse négative.
   - Existe t-il une réfutation linéaire input de l'ensemble de clauses $BUT_2 \cup E_2$ de départ BUT_2 et d'input E_2 ? Si oui, en donner une, sinon, justifiez votre réponse négative.

3. Est-il vrai que si $\mathcal{R}$ est une réfutation linéaire input d'un ensemble de clauses $BUT \cup E$ où BUT Est une clause *négative* et E est un ensemble de clauses *définies*, alors tout noeud interne est une clause négative ? Justifiez votre réponses.

Correction :

1. *Voici une réfutation linéaire input de $BUT_1 \cup E_1$; ci-dessous, on indique, dans l'ordre, les clauses noeuds internes $cl_1, cl_2, cl_3, cl_4, cl_5$:*

   - *cl_1 est*
   $$\neg move(a, b, c, suc(0), Output)$$
   obtenue par RES à partir de $\neg hanoi(succ(0), Z)$ $hanoi(N, Output) \ \vee\ \neg move(a, b, c, N, Output)$ via le mgu $\{N \rightarrow suc(0), Z \rightarrow Output\}$

- cl_2 est

$$\neg move(a,c,b,0,Ou1) \ \vee \ \neg move(c,b,a,M,Ou2) \ \vee \ \neg append(Ou1, cons(to(a,b),Ou2), Output)$$

obtenue par RES à partir de cl_1 et un rennomage (où Out est renplacé par $Output'$) de la clause (b) de E_1 via le mgu $\{A \to a, B \to b, C \to c, M \to suc(0), Output' \to Output\}$.

- cl_3 est

$$\neg move(c,b,a,0,Ou2) \ \vee \ \neg append(nil, cons(to(a,b),Ou2), Output)$$

obtenue par RES à partir de cl_2 et $move(A,B,C,0,nil)$ via le mgu $\{A \to c, B \to b, A \to c, Ou1 \to nil\}$

- cl_4 est

$$\neg(append(nil, cons(to(a,b),nil), Output)$$

obtenue par RES à partir de cl_3, et, à nouveau,

$$move(A,B,C,0,nil)$$

via le mgu $\{A \to c, B \to b, C \to a, Ou2 \to nil\}$

- De cl_4 et $append(nil,L1,L1)$ on obtient, par RES, la clause vide (cl_5), via le mgu $\{L1 \to cons(to(a,b),nil), Output \to cons(to(a,b),nil)\}$

La clause $append(cons(A,L1),L2,cons(A,L3)) \ \vee \ \neg append(L1,L2,L3)$ de E_1 n'est jamais utilisée.

2.
   - N'existe pas : $FACT$ est nécessaire pour aboutir à la clause vide. Remarquez que le nombre de fois où r apparaîtrait dans les noeuds internes d'une dérivation n'utilisant pas $FACT$ est égal à 2.
   - Par contre, on obtient facilement une réfutation de $BUT_2 \cup E_2$qui n'est pas linéaire en appliquant d'abord $FACT$ à une des deux clauses de E_2 :

$$\frac{\neg r(y,v,f(a)) \vee \ \neg r(g(a),v,f(v))}{\neg r(g(a),a,f(a))} \ FACT$$

   puis en l'applicant à l'autre clause de E_2 :

$$\frac{r(x,a,f(z)) \ \vee \ r(g(w),w,f(w)) \ \vee s(a)}{r(g(a),a,f(a)) \vee s(a)} \ FACT$$

   en appliquant RES aux deux rèsultats ainsi obtenus :

$$\frac{\neg r(g(a),a,f(a)), \ r(g(a),a,f(a)) \vee s(a)}{s(a)} \ RES$$

   et, enfin, en applicant RES à $s(a)$ et à BUT_2.

3. Oui. Soient $c_1,...,c_n$ les noeud internes d'une dérivation linéaire. Puisque c_1 est la conclusion de l'inférence la plus haute, ayant la forme

$$\frac{BUT \quad cl}{c_1} \ RES$$

où BUT est négative et cl est une clause définie, forcement tout littéral de c_1 est négatif. De même, en sachant que c_i $(i < n)$, est une clause négative, puisque c_{i+1} est la conclusion d'une inférence de la forme :

$$\frac{c_i \quad cl}{c_{i+1}} \ RES$$

où cl est une clause définie, forcement c_{i+1} est une clause négative.

Exercice 4
Barème indicatif : 3 points.
Soit $\mathcal{L}$ un langage du calcul des prédicats du premier ordre dont les seuls symboles de prédicat sont $p_1, ..., p_k$ $(k > 0)$, à 1 argument, et ne contenant pas de symboles de fonction. Soit $\mathcal{I}$ une interprétation de $\mathcal{L}$, ayant D comme domaine. On dit que deux éléments e_1 et e_2 de D sont *distingables* s'il existe au moins une formule A de $\mathcal{L}$ avec n variables libres $x_1, ..., x_n$ $(n > 0)$ et telle que, pour au moins un $(n-1)$-uplet $\langle d_1, ..., d_{n-1}\rangle$ d'éléments de D, A est vraie pour le choix de valeurs $\langle x_1 := d_1, ..., x_{n-1} := d_{n-1}, x_n := e_1\rangle$ mais elle est fausse pour $\langle x_1 := d_1, ..., x_{n-1} := d_{n-1}, x_n := e_2\rangle$. Montrez que si e_1 et e_2 sont distingables, alors e_2 et e_1 sont distingables. Donner un majorant du nombre d'éléments distingables de D. En déduire l'existence d'un algorithme permettant de décider la validité d'une formule quelconque de $\mathcal{L}$. Justifiez vos réponses, mais de façon *très synthétique.*

Correction : *Supposons que e_1 et e_2 soient distingables. Par définition, il existe au moins une formule A de $\mathcal{L}$ avec n variables libres $x_1, ..., x_n$ $(n > 0)$ et telle que, pour au moins un $(n-1)$-uplet $\langle d_1, ..., d_{n-1}\rangle$ d'éléments de D, A est vraie pour le choix de valeurs $\langle x_1 := d_1, ..., x_{n-1} := d_{n-1}, x_n := e_1\rangle$ mais elle est fausse pour $\langle x_1 := d_1, ..., x_{n-1} := d_{n-1}, x_n := e_2$. La formule $\neg A$ montre alors que e_2 et e_1 sont aussi distingables.*

Pour que deux éléments e_1 et e_2 soient distingables, il faut et il suffit que l'interprétation d'au moins un prédicat p_i diffère entre l'un et l'autre. Comme il y a 2^k valeurs de vérité possibles pour l'ensemble de p_i, il y a au maximum 2^k éléments distingables.

Par conséquent, pour décider de la validité d'une formule quelconque A de L, il suffit de vérifier la validité de la formule A pour les interprétations dont le domaine a au plus de 2^k éléments. Sans perte de genéralité, on peut supposer que, si $1 \geq n \geq 2^k$, un domaine avec n valeurs est exactement l'ensemble de naturels $\{1, ..., n\}$. Par conséquent, l'ensemble d'interprétations à considerer pour tester la validité de A est fini, donc il existe un algorithme qui les énumère toutes et vérifie que la formule est vraie pour chacune. Inutile de dire que cet algorithme est d'une complexité pornographique!